LINGUAGEM UML
INTRODUÇÃO

Luís Morgado

2023

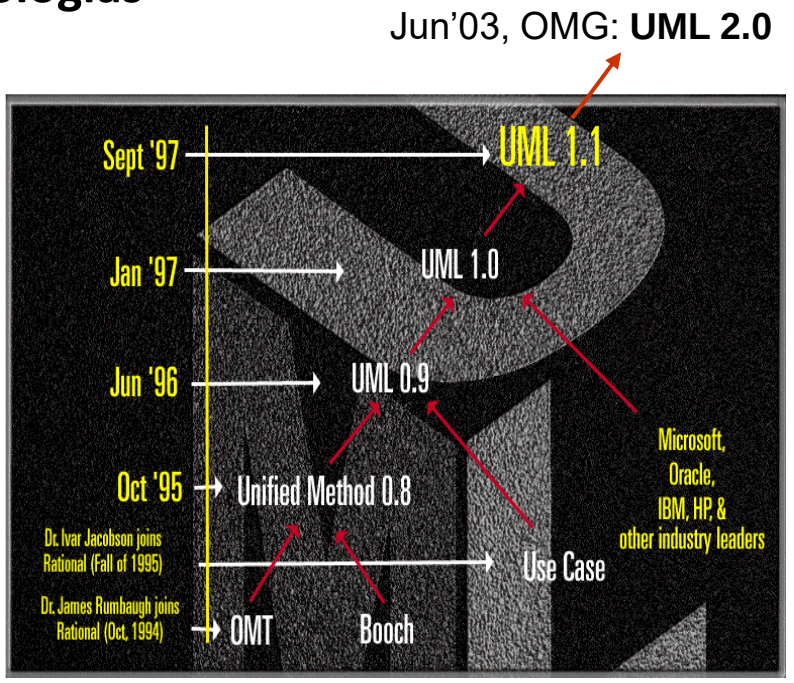
INTELIGÊNCIA ARTIFICIAL E ENGENHARIA DE SOFTWARE

- O desenvolvimento de sistemas com base em inteligência artificial é caracterizado pela elevada complexidade desses sistemas, requerendo métodos adequados de engenharia de software, nomeadamente, no que se refere à modelação de sistemas, bem como a linguagens de modelação adequadas, como é o caso da linguagem UML
- A linguagem UML (*Unified Modeling Language*) é uma linguagem de modelação geral de sistemas, orientada em particular para a modelação de software
- Vantagens:
  - Orientada para a concepção e modelação de software de forma organizada e sistemática, facilitando a compreensão e descrição dos modelos desenvolvidos
  - É uma linguagem normalizada que facilita a compreensão e comunicação dos modelos, quer para quem concebe os modelos, quer para quem os utiliza
  - Facilita a produção de código, possibilitando uma tradução organizada e sistemática dos modelos para código, incluindo a geração automática de código
  - Adopta uma abordagem *orientada a objectos*, o que contribui para lidar com a complexidade, bem como para facilitar a reutilização de código

LINGUAGEM UML

UML - *Unified Modeling Language*

- **Resulta da combinação de várias metodologias** de modelação orientadas a objectos
  - OMT (*Jim Rumbaugh*)
  - Booch Method (*Grady Booch*)
  - OOSE (*Ivar Jacobson*)
- **Linguagem para descrever conhecimento**
  - Acerca do domínio do problema
  - Acerca do domínio da solução
  - Linguagem gráfica
    - Elementos gráficos e textuais

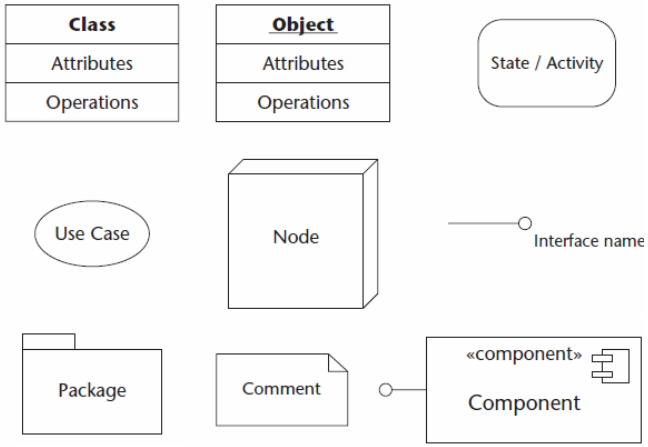


[Rational]

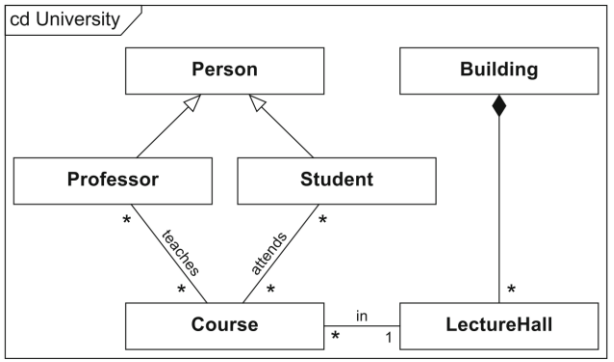
- **Normalizada no âmbito do OMG** (*Object Management Group*)
  - Especificações formais
  - Adequada a diferentes âmbitos de modelação de sistemas

LINGUAGEM UML

- **Classificadores** (tipos de elementos base)
  - *Class*
  - *Association*
  - *Interface*
  - *Component*
  - *Package*
  - *State*
  - *Activity*
  - (…)



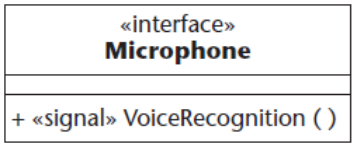
- **Diagramas**
  - Na linguagem UML os modelos são organizados em diagramas que permitem representar diferentes perspectivas de modelação de um sistema
  - Representação gráfica facilita a percepção das relações entre elementos



LINGUAGEM UML

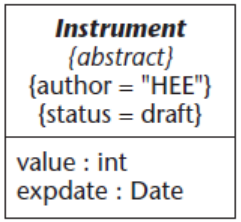
- **Mecanismos de Extensão**
  (*Extensibility Mechanisms*)
  - **Estereótipos** (*Stereotypes*)
    - Mecanismo de extensão da linguagem que possibilita a atribuição de significado específico a elementos base da linguagem
    - Permitem estender o vocabulário UML de modo a definir elementos de modelação, com características específicas, a partir dos já existentes
    - Podem ter representação textual ou gráfica
  - **Propriedades** (*Tagged values*)
    - São utilizadas para definir propriedades dos elementos da linguagem, com eventual associação de informação específica, por exemplo, autor ou versão
  - **Restrições** (*Constraints*)
    - Especificam a semântica ou condições que se devem verificar em relação a elementos de um modelo

Representação textual de estereótipo
no formato <<*estereótipo*>>

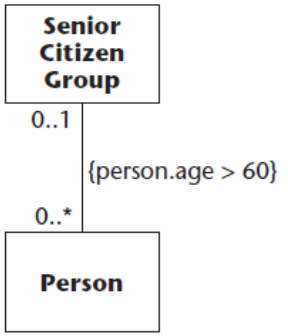


Notação:
«*estereótipo*»

Representação gráfica
(estereótipo *interface*)



Notação:
{*propriedade*}

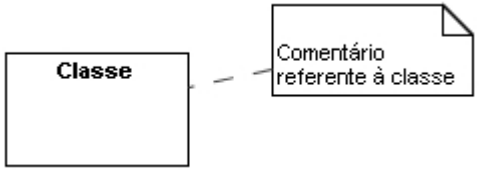


Notação:
{*restrição*}

[Eriksson et al., 2004]

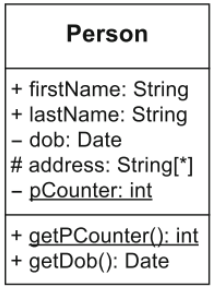
LINGUAGEM UML

- **Complementos de Informação**
  - **Adornos**
    - Informação textual ou gráfica indicativa de aspectos específicos de um elemento, por exemplo, atributo ou método *estático*
  - **Comentários**
    - Elemento gráfico contendo informação textual referente a elementos de um modelo



Nome de classe: **Negrito**
Nome de objecto: **Sublinhado**

Atributo ou método estático: Sublinhado



[Seidl, 2012]

LINGUAGEM UML

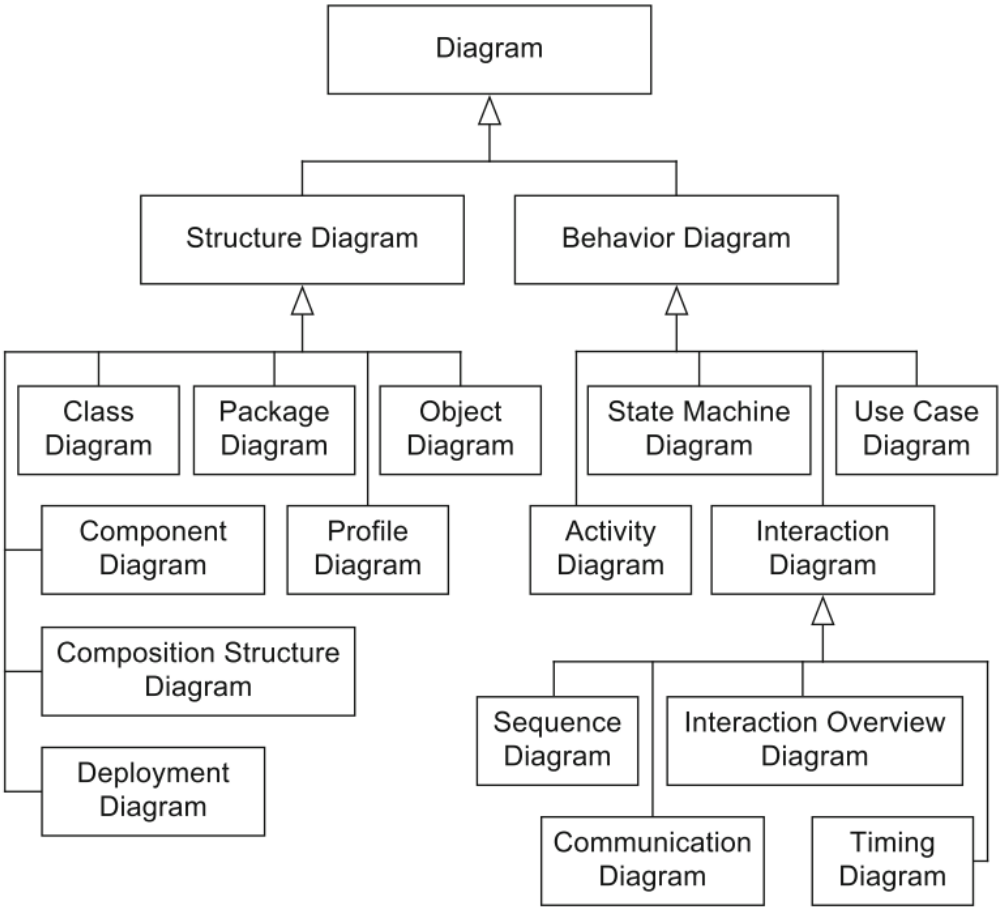
PERSPECTIVAS DE MODELAÇÃO

Diagramas

Na linguagem UML os modelos são organizados em diagramas que permitem representar diferentes perspectivas de modelação de um sistema, organizados em duas perspectivas principais:

- ***Perspectiva estrutural***, referente à organização das partes e relações entre partes que constituem a estrutura de um sistema
- ***Perspectiva comportamental***, referente à organização funcional e dinâmica de um sistema que determina o seu comportamento

Tipos de diagramas

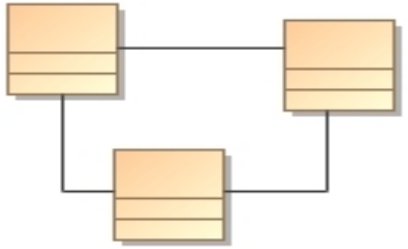


[Seidl, 2012]

DIAGRAMAS DE CLASSES

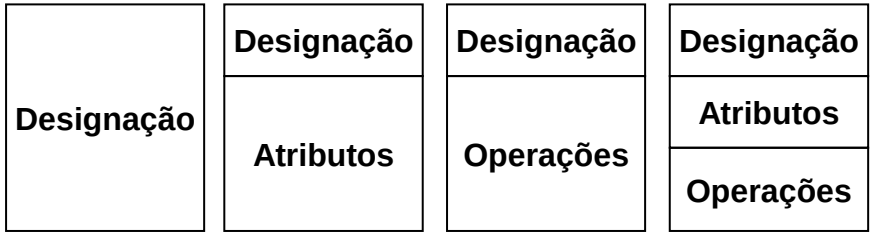
REPRESENTAÇÃO DE ESTRUTURA

- **Descrevem uma abstracção das partes e das relações entre partes de um sistema**
  - Organização estática do sistema
  - Foco na estrutura

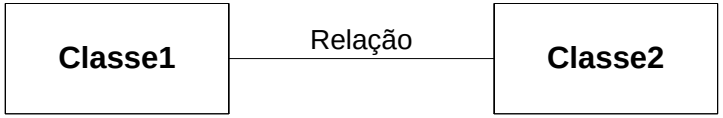


- **Classes**
  - Atributos
    - Definição de estrutura
  - Operações (*métodos*)
    - Definição de comportamento

Diferentes modos de representar graficamente uma classe, com diferentes níveis de detalhe

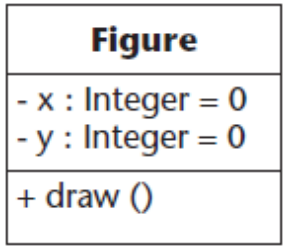


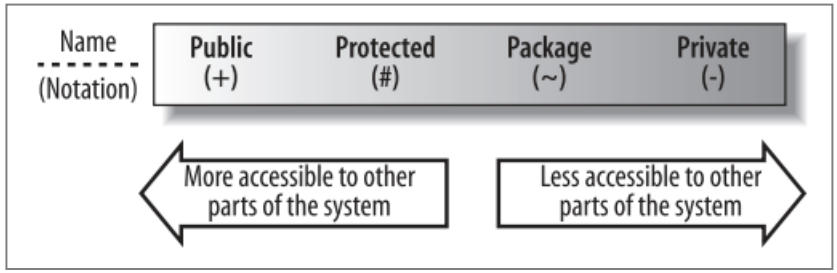
- **Relações**
  - Representam interdependência entre partes



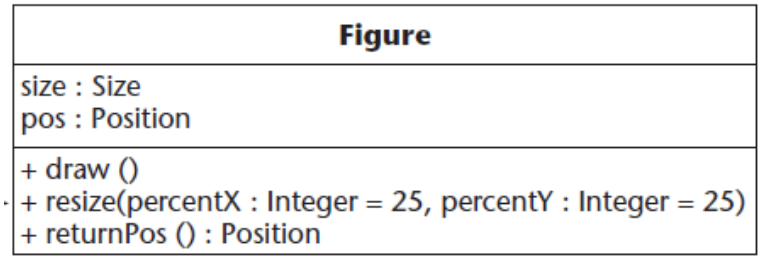
DIAGRAMAS DE CLASSES

- **Atributos**
  - Representação de estrutura
    - Elementos de informação
  - Caracterizados por:
    - Designação
    - Tipo
    - Visibilidade
      - Público (+)
      - Privado (-)
      - Protegido (#)
      - Pacote (~)
  - Sintaxe:
    - visibility name:type = init_value {property_string}
- **Operações**
  - Representação de comportamento
    - Acções ou funções suportadas



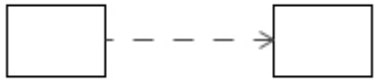


[Miles & Hamilton, 2006]

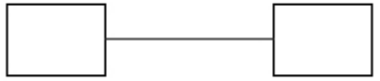


DIAGRAMAS DE CLASSES

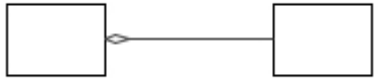
- **Relações** entre classes
  - Dependência
    - Relação na qual uma parte utiliza ou depende de outra parte, por exemplo, uma classe tem um método com um parâmetro que é uma instância de outra classe, ou cria localmente uma instância de outra classe



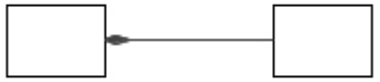
  - Associação
    - Relação na qual uma parte está estruturalmente associada a outra parte através de um dos seus atributos, ou seja, um dos seus atributos tem informação acerca de outra parte, por exemplo, uma classe tem um atributo cujo tipo é uma instância de outra classe



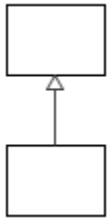
  - Agregação
    - Caso particular de associação que indica que uma parte agrega outras partes, as quais podem existir independentemente da parte agregadora, por exemplo, a relação entre turma e aluno



  - Composição
    - Caso particular de associação que indica que uma parte é composta por outras partes que só existem no contexto da parte composta, por exemplo, a relação entre apartamento e divisão (o apartamento é composto por várias divisões que não existem separadas do todo)



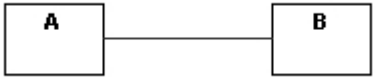
  - Generalização
    - Relação estrutural que indica que uma parte é uma especialização de outra parte mais geral



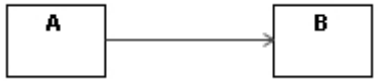
DIAGRAMAS DE CLASSES

- **Propriedades** das relações
  - Direcção
    - Indica a navegabilidade de uma associação, ou seja, que partes contêm informação acerca de outras partes
    - Pode ser *bidirecional* ou *unidirecional*
  - Multiplicidade
    - Indica quantas instâncias de uma parte estão associadas a instâncias de outras partes
  - Papel
    - Indica o papel de uma parte numa associação, ou seja, o significado concreto ou a responsabilidade que lhe corresponde
  - Qualificação
    - O qualificador de uma associação designa uma chave utilizada para obter um item da colecção respectiva

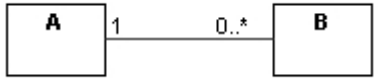
Associação *bidirecional*, as instâncias de ambas as partes conhecem-se reciprocamente



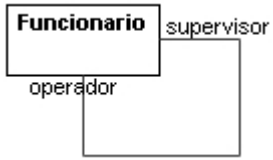
Associação *unidirecional*, apenas as instâncias da parte A conhecem instâncias da parte B



Por cada instância de A existem zero ou mais de B, por cada instância de B existe uma instância de A



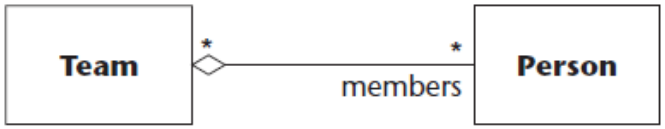
As instâncias de funcionário desempenham os papéis de *operador* ou *supervisor* (cada operador tem associado um supervisor, cada supervisor tem associado um operador



O acesso aos produtos de uma encomenda é realizado através do atributo *codigoProduto*

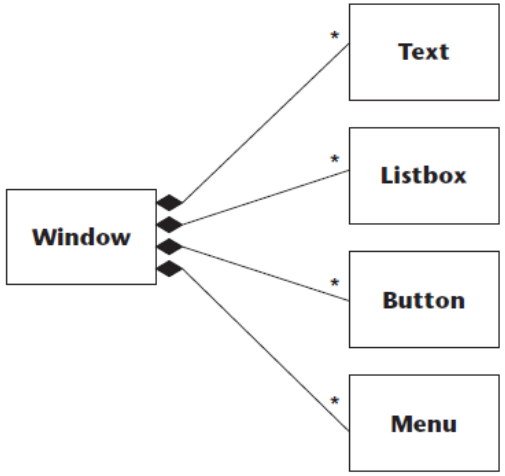
DIAGRAMAS DE CLASSES

- **Agregação**
  - Relação Parte - Todo
  - Denota que uma parte contém outra em termos lógicos ou físicos
  - Forma fraca de composição de partes
    - Não define restrições semânticas acerca de:
      - Pertença das partes
      - Criação e destruição das partes
      - Períodos de existência das partes



Relação *parte-todo*

- **Composição**
  - Forma forte de composição de partes
    - Define restrições semânticas específicas
      - O todo cria e destrói as partes
      - Sobreposição de períodos de existência das partes
      - Implica exclusividade na composição das partes
    - Organização hierárquica em árvore
      - Níveis de abstracção

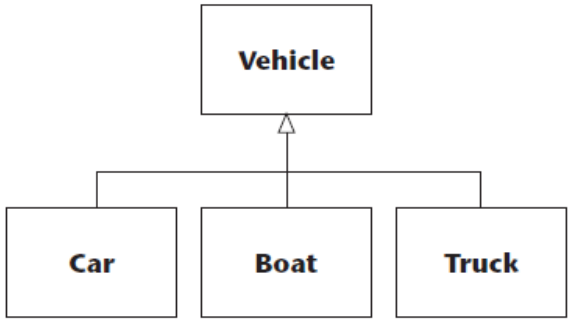


Regra da *não-partilha*

[Eriksson et al., 2004]

DIAGRAMAS DE CLASSES

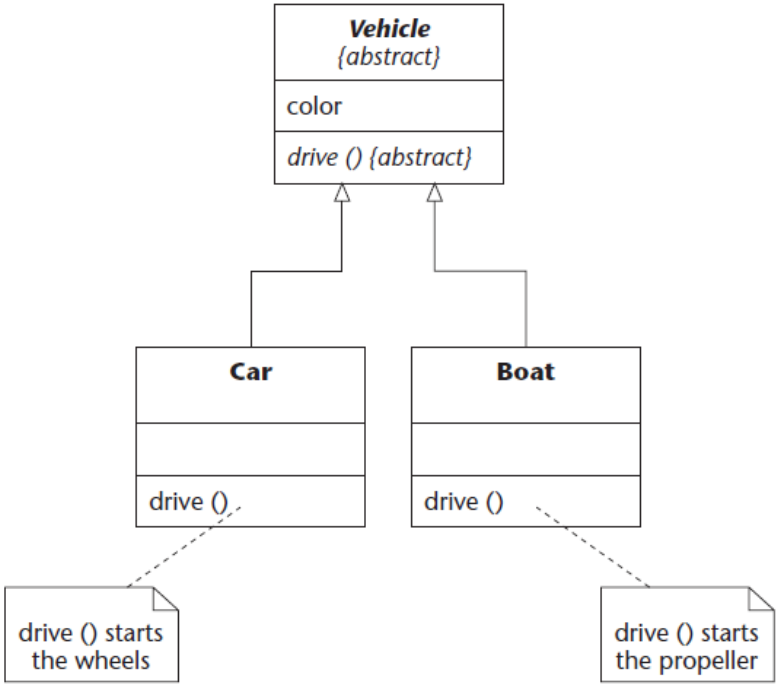
- **Generalização**



[Eriksson et al., 2004]

A *generalização* é uma relação estrutural que indica que uma parte é uma especialização de outra parte mais geral

- **Polimorfismo**



O *polimorfismo* é uma característica da modelação e programação orientada a objectos que possibilita que diferentes tipos de objectos, com operações comuns, tenham diferentes implementações dessas operações que podem ser utilizadas de forma homogénea independentemente do tipo dos objectos

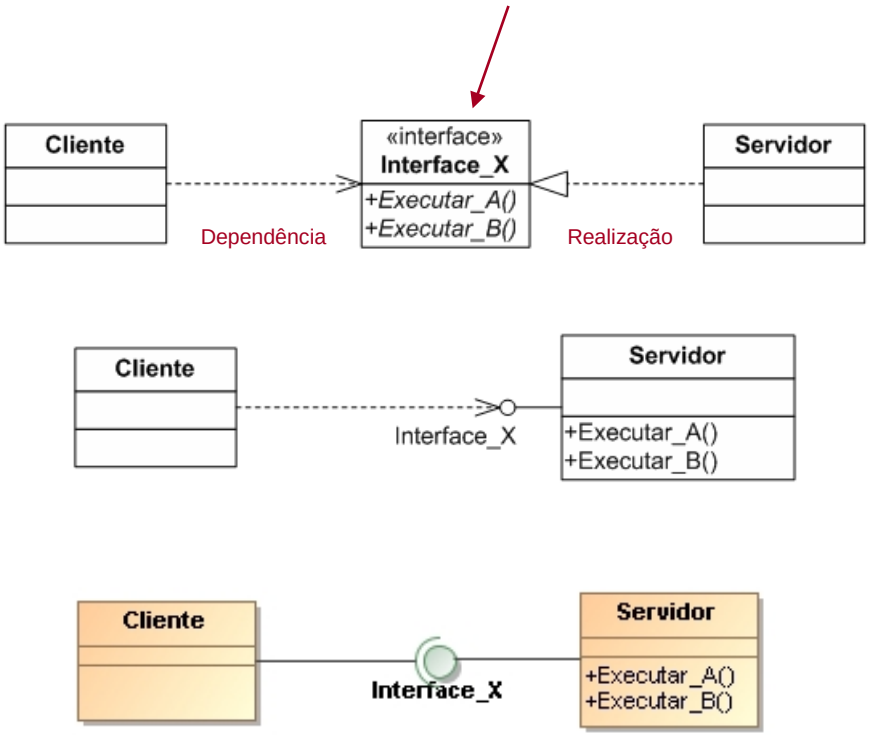
Isso é conseguido com uma definição comum das operações, por exemplo, através de uma *classe abstracta* ou de uma *interface*

DIAGRAMAS DE CLASSES

- **Interface**
  - Classificador que define as características visíveis de uma classe
  - Conjunto **coeso** de características
  - Define um contrato
    - Não implementa essas características
  - Promove a modularidade através do **encapsulamento** da implementação
    - Define um contrato de serviços independente da forma de implementação

- **Realização**
  - Representa implementação
  - Relação entre uma interface e uma classe ou um componente
  - Uma classe ***realiza*** as características de uma interface implementando-as
  - Uma classe pode implementar mais que uma interface

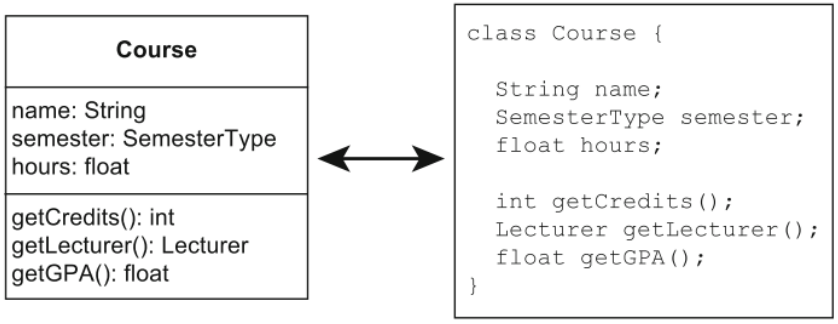
A interface define um contracto de prestação de serviços independente da implementação



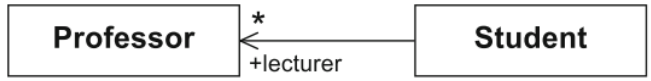
Diferentes notações gráficas associadas ao conceito de interface

DIAGRAMAS DE CLASSES

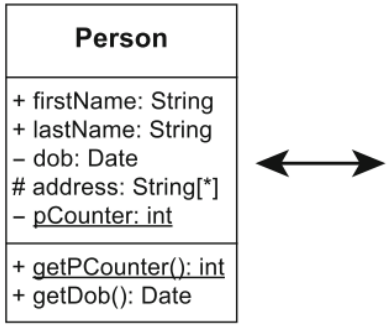
Exemplo de tradução para código (linguagem Java)




```
class Course {

  String name;
  SemesterType semester;
  float hours;

  int getCredits();
  Lecturer getLecturer();
  float getGPA();
}
```




```
class Professor {…}

class Student {
  public Professor[] lecturer;
  …
}
```




```
class Person {

  public String firstName;
  public String lastName;
  private Date dob;
  protected String[] address;
  private static int pCounter;

  public static int getPCounter() {…}
  public Date getDob() {…}
}
```

[Seidl, 2012]

Numa associação, uma parte está estruturalmente associada a outra parte através de um dos seus atributos, ou seja, um dos seus atributos tem informação acerca de outra parte, por exemplo, uma classe tem um atributo cujo tipo é uma instância de outra classe

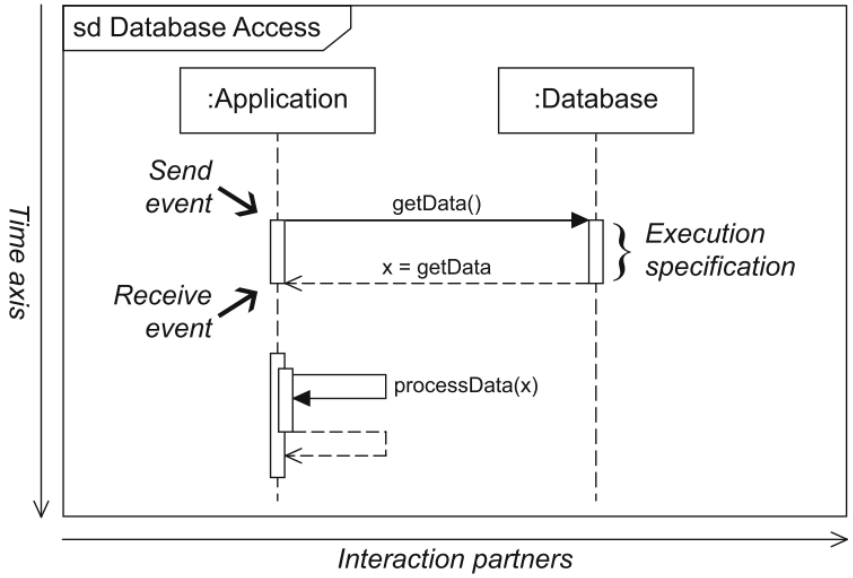
A parte que referencia deve ter um atributo correspondente à parte referenciada, o nome do atributo deve corresponder ao papel da parte referenciada na associação

Se a multiplicidade for diferente de 1, deve ser definido um contentor de elementos correspondente à multiplicidade e eventuais restrições de dados definidas

DIAGRAMAS DE SEQUÊNCIA

REPRESENTAÇÃO DE COMPORTAMENTO

- **Modelo de interacção**
  - Descrevem a comunicação entre partes do sistema e/ou com o exterior
  - Ênfase na sequência temporal de interacção
- **Organização bidimensional**
  - **Tempo** – vertical
  - **Estrutura** (partes) – horizontal
- **Elementos de modelação**
  - **Linha de vida** (*lifeline*)
    - Representa evolução temporal
  - **Foco de activação** (*activation bar*)
    - Representam execução de operações
  - **Mensagem**
    - Partes trocam mensagens
  - **Operador**
    - Fragmento de interacção com semântica específica

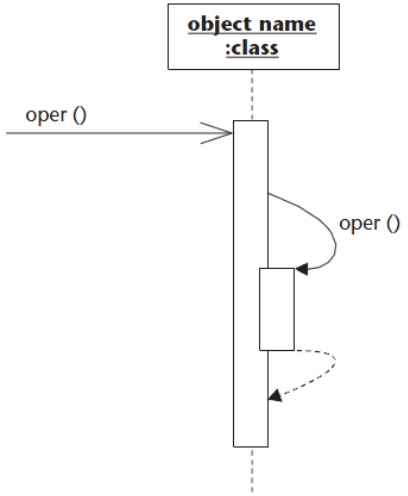


[Seidl, 2012]

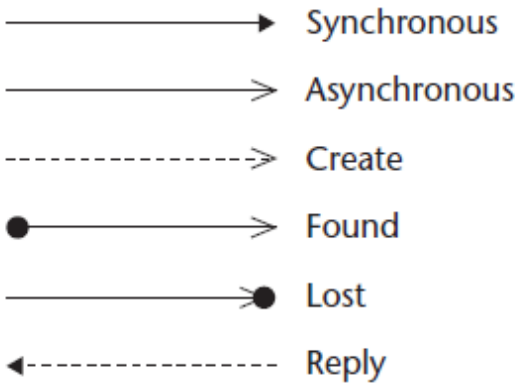
DIAGRAMAS DE SEQUÊNCIA

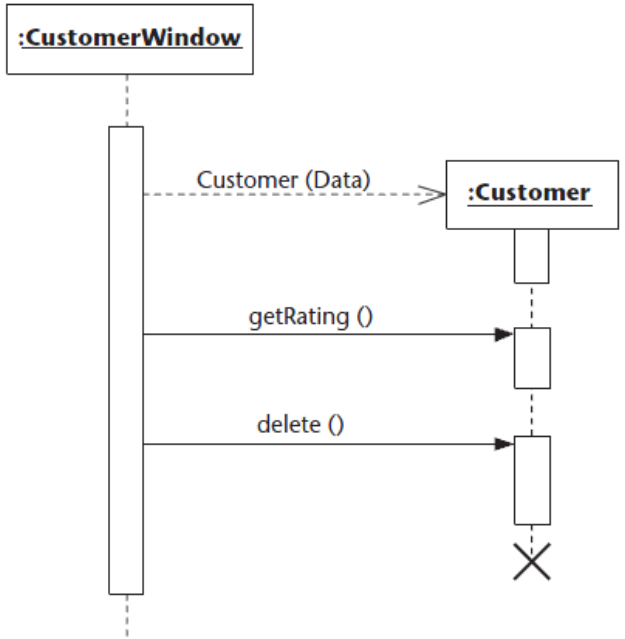
- **Mensagens**
  - **Sintaxe**
    - [attribute=] message-name [(argument)] [:return-value]

[Condição] Mensagem



- **Tipos de mensagens**





DIAGRAMAS DE SEQUÊNCIA

Exemplo: Caso prático – iniciar jogo

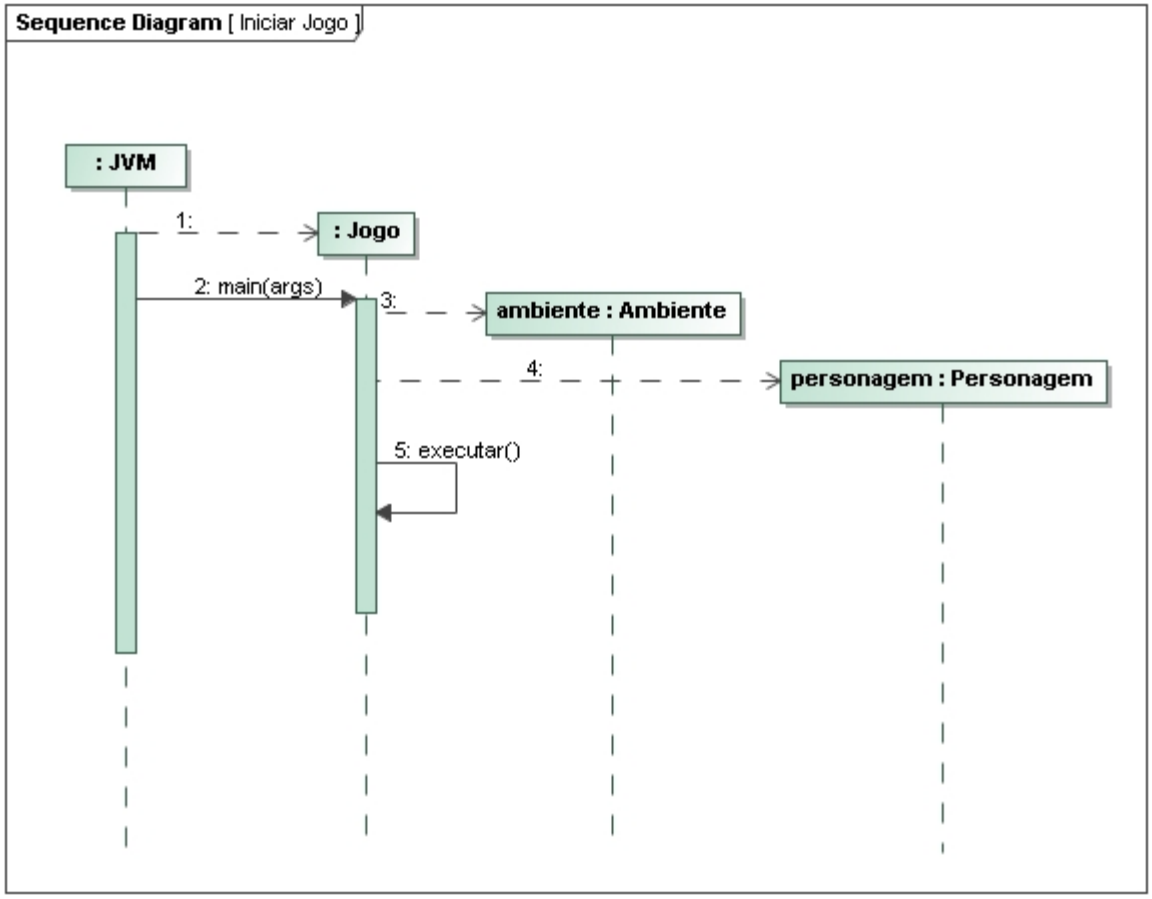
O **jogo** consiste num **ambiente** onde a **personagem** tem por objectivo registar a presença de animais através de fotografias.

Conceitos do domínio do problema

Jogo
- Ambiente
- Personagem

Ambiente de execução

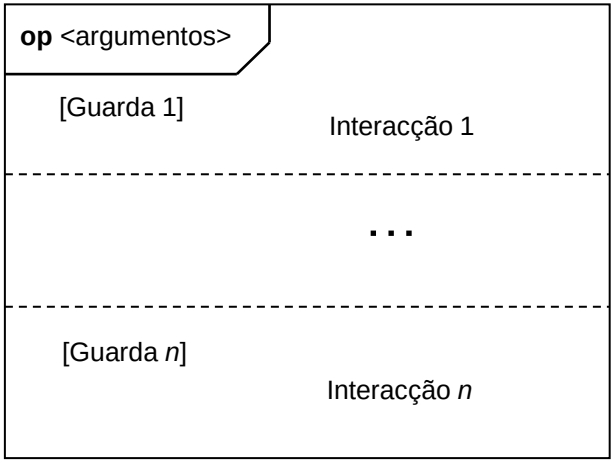
JVM: Java Virtual Machine

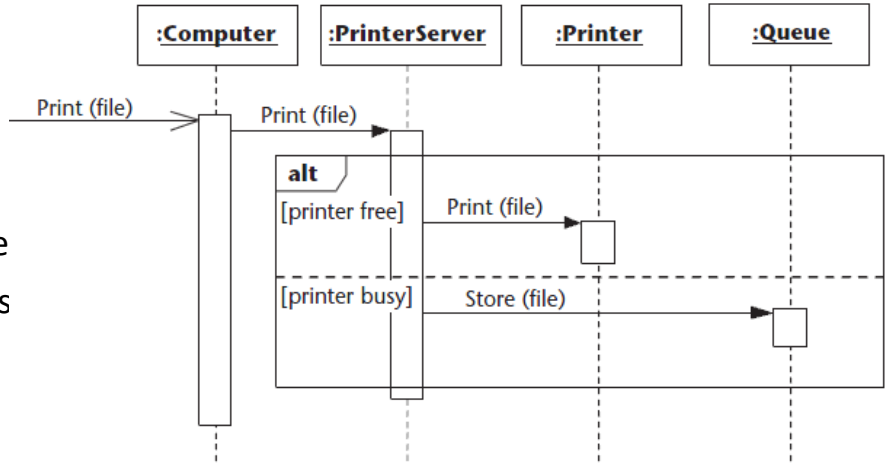


DIAGRAMAS DE SEQUÊNCIA

- **Operador**
  - Fragmento de interacção com semântica específica

- **Tipos de operadores**
  - **ref**: referência a fragmento de interacção
  - **loop**: repetição de fragmento de interacção
  - **break**: fim de repetição de fragmento de interacção
  - **alt**: selecção de fragmento de interacção
  - **par**: regiões concorrentes (paralelas)
  - **assert**: fragmento de interacção requerido
  - **opt**: fragmento de interacção opcional
  - **neg**: especificação negativa (não pode acontece
  - **region**: região crítica (não são permitidas outras mensagens)

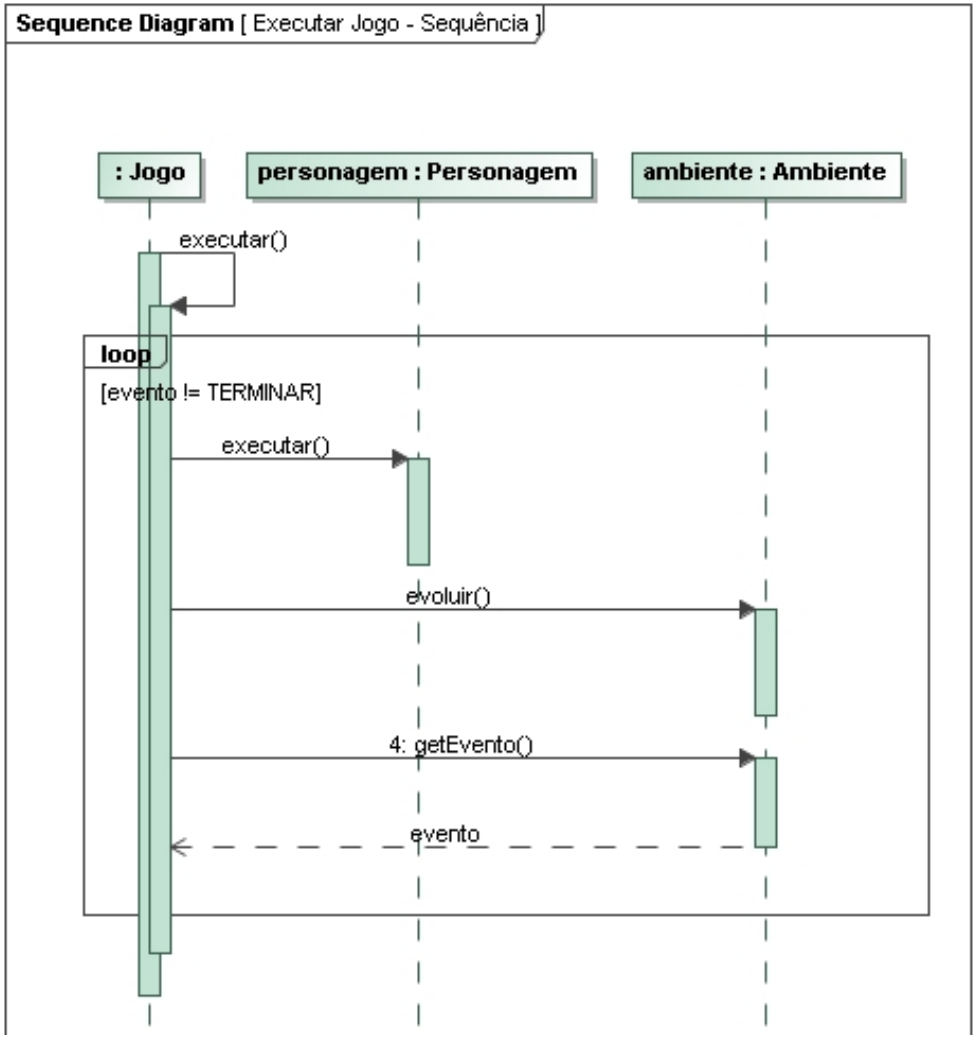




DIAGRAMAS DE SEQUÊNCIA

Exemplo: Caso prático – executar jogo

Operador loop:
repetição de fragmento de interacção



BIBLIOGRAFIA

[Watson, 2008]

Andrew Watson, *Visual Modeling: past, present and future*, OMG, 2008.

[Meyer, 1997]

B. Meyer, *UML: The Positive Spin*, American Programmer - Special UML issue, 1997.

[Yelland et al., 2002]

Yelland, M. J., B. I. Moat, R. W. Pascal and D. I. Berry, *CFD model estimates of the airflow over research ships and the impact on momentum flux measurements*, Journal of Atmospheric and Oceanic Technology, 19(10), 2002.

[Selic, 2003]

B. Selic, *Brass bubbles: An overview of UML 2.0*, Object Technology Slovakia, 2003.

[Graessle, 2005]

P. Graessle, H. Baumann, P. Baumann, *UML 2.0 in Action*, Packt Publishing, 2005.

[Eriksson et al., 2004]

H. Eriksson, M. Penker, B. Lyons, D. Fado, *UML 2 Toolkit*, Wiley, 2004.

[Seidl, 2012]

UML Classroom: An Introduction to Object-Oriented Modeling, M. Seidl *et al.,* Springer, 2012

[Douglass, 2006]
B. Douglass, *Real-Time UML*, Telelogic, 2006.